(S) 739 200

1

Ref only

(5) 739200

(6) A SIMPLE ALGORITHM FOR FINDING MAXIMAL NETWORK FLOWS AND AN APPLICATION TO THE HITCHCOCK PROBLEM,

(10) by L. R. Ford, Jr. and D. R. Fulkerson.

(14) Rept no. P-743 ✓ [illegible]

Revised 2[illegible] December [illegible]

Copy 1

23)

## SUMMARY

A very simple algorithm for finding a maximal flow and minimal cut in a transportation network is described; it is then applied to obtain an efficient computational routine for the Hitchcock distribution problem.

# A SIMPLE ALGORITHM FOR FINDING MAXIMAL NETWORK FLOWS AND AN APPLICATION TO THE HITCHCOCK PROBLEM

L. R. Ford, Jr.
D. R. Fulkerson

## INTRODUCTION

The network-flow problem, originally posed by T. Harris of The RAND Corporation, has been discussed from various view-points in [1], [2], [3], [illegible], [illegible]. The problem arises naturally in the study of transportation networks; it may be stated in the following way. One is given a network of directed arcs and nodes with two distinguished nodes, called <u>source</u> and <u>sink</u>, respectively.[1] All other nodes are called <u>intermediate</u>. Each directed arc in the network has associated with it a nonnegative integer, its <u>flow capacity</u>. Source arcs may be assumed to be directed away from the source, sink arcs into the sink. Subject to the conditions that the flow in an arc is in the direction of the arc and does not exceed its capacity, and that the total flow into any intermediate node is equal to the flow out of it, it is desired to find a maximal flow from source to sink in the network, i.e., a flow which maximizes the sum of the flows in source (or sink) arcs.

---

[1] A problem in which there are several sources and sinks, with flows permitted from any source to any sink, is reducible to a single-source, single-sink problem.

For example, consider the network of Fig. 1 with

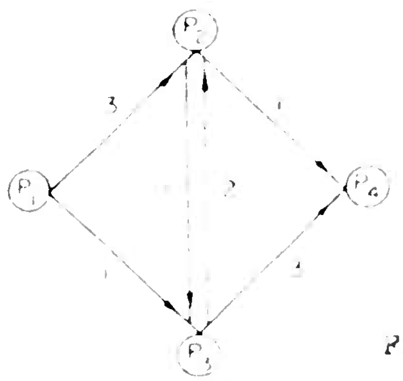

Fig. 1

source $P_1$, sink $P_4$, and arc capacities as indicated. If we let $x_{ij}$ denote the flow from $P_i$ to $P_j$, the problem is to maximize $x_{12} + x_{13}$, the total flow leaving $P_1$, subject to the equations and inequalities

$$x_{12} + x_{32} = x_{23} + x_{24},$$

$$x_{13} + x_{23} = x_{32} + x_{34},$$

$$0 \leq x_{12} \leq 3,$$

$$0 \leq x_{13} \leq 1,$$

$$0 \leq x_{23} \leq 1,$$

$$0 \leq x_{32} \leq 2,$$

$$0 \leq x_{24} \leq 1,$$

$$0 \leq x_{34} \leq 3.$$

In general, if we let $P_1$ be the source, $P_n$ the sink, we are required to find $x_{ij}(i,j = 1,\dots n)$ which maximize

(i) $$\sum_{j=2}^{n} x_{1j}$$

subject to

(2) $$\sum_j (x_{ij} - x_{ji}) = 0 \qquad (i = 2,\ldots,n-1) ,$$

$$0 \leq x_{ij} \leq c_{ij} ,$$

where the $c_{ij}$ are given nonnegative integers, and, in particular, $c_{i1} = c_{nj} = 0$ for all i, j.

This is of course a linear programming problem, and hence may be solved by Dantzig's simplex algorithm. In fact, the simplex computation for a problem of this kind is particularly efficient, since it can be shown that the sets of equations one solves in the process are always triangular [2]. However, for the flow problem, we shall describe what appears to be a considerably more efficient algorithm; it is, moreover, readily learned by a person with no special training, and may easily be mechanized for handling large networks. We believe that problems involving more than 500 nodes and 4,000 arcs are within reach of present computing machines.

Of some theoretical interest is the fact that the procedure assures one of obtaining a strict increase in the total flow at each step (in contrast with the simplex method). In addition, the Hitchcock problem can be solved via the flow algorithm in a way which naturally generalizes the combinatorial method recently proposed by Kuhn [7] for the optimal-assignment problem. Informal tests by hand indicate that this way of solving the Hitchcock problem is extremely efficient.

## 1. COMPUTATION OF MAXIMAL FLOW, AN EXAMPLE

In order to illustrate the computational procedure, before describing it in general terms, let us return to the example of Fig. 1. Start out with any flow from source to sink, say a flow of 1 along the chain $P_1$ $P_3$ $P_2$ $P_4$. We describe this flow by Fig. 2 below,

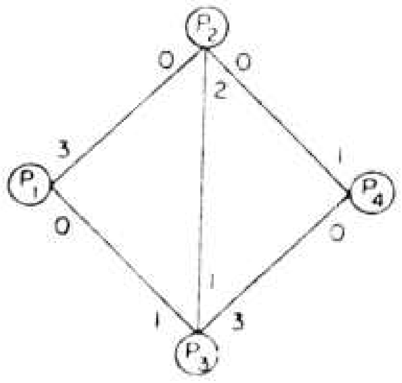

Fig. 2

where the remaining or unused capacity from $P_i$ to $P_j$ is denoted by the number closest to $P_i$ on the arc $P_i$ $P_j$. Notice, for example, that the new capacity from $P_3$ to $P_2$ has been decreased by 1, and the capacity from $P_2$ to $P_3$ has been increased by 1. That is to say, we must now allow the possibility, in constructing successive flows, of imposing a flow of 2 from $P_2$ to $P_3$, one unit of which would cancel the present flow of 1 from $P_3$ to $P_2$. Now start with the source and look for nodes which may be reached "in one step" by arcs of strictly positive remaining capacity. Here we may proceed only to $P_2$. When a node has been reached, label it in some fashion; then take a labeled node not previously examined, and look for unlabeled nodes which may be reached in one step by arcs of positive remaining capacity. Here we may proceed from $P_2$ only to $P_3$. Repeat the procedure until either (a) the sink has been labeled, or (b) no further nodes may be labeled and the sink has not been labeled. In our example, case (a) occurs at the next step. We may now search back[*] through labeled nodes, locating a chain from source to sink along which an additional flow may be imposed. Here we have the chain $P_1P_2P_3P_4$, and a flow of 2 may be imposed. We thus obtain Fig. 3 below

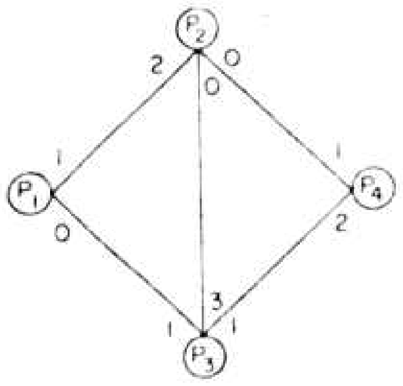

Fig. 3

from Fig. 2 by alternately subtracting and adding 2 to the numbers encountered along the chain $P_1P_2P_3P_4$. Next repeat the labeling procedure. This time we can label only $P_1$ and $P_2$, and are thus in case (b). This means that we have obtained a maximal flow, depicted in Fig. 4.

[*] In the general description of the algorithm to be given presently, we will carry enough information along to make the backward search unnecessary.

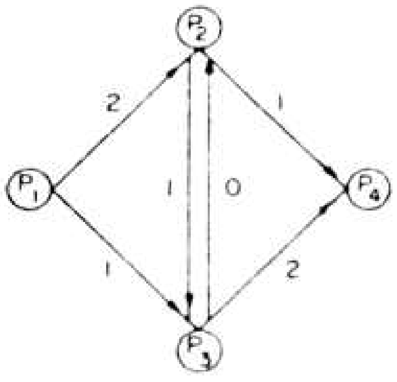

Fig. 4

The proof that this flow is maximal in this case is immediate, for observe that the directed arcs $P_1P_3$, $P_2P_3$, $P_2P_4$ leading from labeled nodes to unlabeled nodes are all saturated and form a cut in the network; i.e., every directed chain from source to sink contains one of these directed arcs.[3] Since it is clear that the sum of capacities of arcs forming a cut, which we refer to as the value of the cut, is an upper bound for flow values, and since we have achieved equality of flow value and a cut value, the flow is maximal and the directed arcs $P_1P_3$, $P_2P_3$, $P_2P_4$ constitute a minimal cut, i.e., a cut of minimal value.

## 2. THE MINIMAL CUT THEOREM

A nonconstructive proof of the minimal cut theorem, which asserts equality of maximal flow value and minimal cut value, has been given by the present writers in [5]. Subsequently a constructive proof based on the simplex criterion of linear

[3]The definition of cut given here corresponds to that of disconnecting set given in [5].

programming was developed [2,3]. The algorithm which we describe more formally in the next section also provides a constructive proof of the theorem. Like the simplex algorithm, it produces not only a maximal flow but a minimal cut as well. This will be important for our application to the Hitchcock problem.

It should perhaps be pointed out that an undirected problem, by which we mean that the directions of flow in intermediate arcs are not specified, so that the capacity constraints on these arcs are of the form

(3) $$x_{ij} + x_{ji} \leq c_{ij} \qquad (i,\ j = 2,\ldots,n-1;\ i<j)\ ,$$

presents nothing new, since we may replace each undirected arc by a pair of oppositely directed arcs, each with capacity equal to that of the original arc; i.e., replace (3) by

(4) $$\begin{aligned} x_{ij} &\leq c_{ij}\ , \\ x_{ji} &\leq c_{ij}\ . \end{aligned}$$

For, given $X' = (x'_{ij})$ satisfying (4) and the conservation equations at intermediate nodes, setting

(5) $$x_{ij} = \max\ (x'_{ij} - x'_{ji}, 0)$$

yields an equivalent flow $X = (x_{ij})$ satisfying these equations and (3).

The minimal cut theorem is true for undirected networks as well as for directed networks, or, more generally, for mixed networks in which some intermediate arcs are directed, others not, the obvious changes in definitions of cuts and chains having been made. This follows from the comments above and the fact, easily proved, that the minimal cut value is the same for a mixed network and its equivalent directed network. The theorem is still valid when capacity constraints on nodes are admitted, where a cut now, of course, includes nodes as well as arcs. This may be proved by splitting each node into two nodes as suggested in Fig. 5

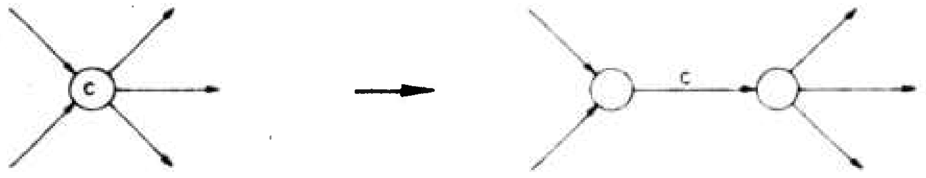

Fig. 5

and placing the capacity c of the old node on the new directed arc joining the two new nodes, thus obtaining an equivalent network with capacity constraints on arcs only. For a direct proof of the theorem in this general form, see [2].

## 3. <u>THE ALGORITHM</u>

We start the computation from any convenient initial flow whatsoever. The initial flow is used to define a starting matrix $A = (a_{ij})$ by letting $a_{ij}$ be the capacity of the arc from $P_i$ to $P_j$ diminished by the flow from $P_i$ to $P_j$ and increased by the flow from $P_j$ to $P_i$. If no other flow is readily available, one may start with the zero flow, corresponding to $A = C$, where $c_{ij}$ is the original capacity from $P_i$ to $P_j$.

(It is the opinion of the authors that if the problem is given in matrix form, no special attempts should be made to obtain a good starting solution. If, on the other hand, the problem can be pictured readily as a linear graph, a "flooding" idea described in [1] might be used to obtain a starting flow. By following the approach suggested in [1], which, however, calls for the exercise of judgment, an initial flow can be obtained that often is optimal for simple networks. If not, it might be used as a good starting point for initiating the procedure given in this paper.)

We are assuming that the notation has been chosen so that $P_1$ is the source, $P_n$ the sink. For certain values of $j = 1 \ldots, n$ we shall define labels $v_j, \mu_j$ recursively as follows.

Let $v_1 = \infty$, $\mu_1 = 0$. For those $j$ such that $a_{1j} > 0$, define $v_j = a_{1j}, \mu_j = 1$. In general, from those $i$ which have received labels $v_i, \mu_i$ but which have not previously been examined, select an $i$ and scan for all $j$ such that $a_{ij} > 0$ and $v_j, \mu_j$ have not been defined. For these $j$, define

(6) $$v_j = \min (v_i, a_{ij}), \mu_j = i.$$

Continue this process until $v_n, \mu_n$ have been defined, or until no further definitions may be made and $v_n, \mu_n$ have not been defined. In the latter case the computation ends. In the former case, proceed to obtain a new $a_{ij}$ matrix as follows.

Replace $a_{\mu_n n}$ by $a_{\mu_n n} - v_n$ and $a_{n \mu_n}$ by $a_{n \mu_n} + v_n$. In general, replace $a_{\mu_j j}$ by $a_{\mu_j j} - v_n$ and $a_{j\mu_j}$ by $a_{j\mu_j} + v_n$, where each $j$ is the $\mu_{j'}$ of the preceding $j'$ in the backward replacement. This replacement continues until $\mu_j = 1$ has been completed. The labels $v_j, \mu_j$ are then recomputed on the basis of the new A matrix and the process is repeated.

Notice that $v_n$ is a positive integer, and hence the process terminates. Upon termination, the maximal flow X is given by defining

$$x_{ij} = \max\,(c_{ij} - a_{ij}, 0), \tag{7}$$

as we now prove.

Lemma 1. X is a flow.

Proof. Clearly $0 \le x_{ij} \le c_{ij}$, since $a_{ij} \ge 0$. It remains to show that X satisfies $\sum_j (x_{ij} - x_{ji}) = 0$ for $i = 2,\ldots,n-1$. Now the process ensures that $c_{ij} + c_{ji} = a_{ij} + a_{ji}$. It follows from this and the definition of X that

$$\sum_j (x_{ij} - x_{ji}) = \sum_j (c_{ij} - a_{ij}) .$$

It therefore suffices to prove that $\sum_j a_{ij}$ is invariant, for $i = 2,\ldots,n-1$, under the computation, as certainly $\sum_j (c_{ij} - a_{ij}) = 0$ for the starting point $A = C$. But if $i$ ($\neq 1,n$) is the $\mu_\ell$ of some $\ell$, then there is a $k = \mu_i$. Thus, for this $i$, the new $a'_{ij}$ are either equal to the old $a_{ij}$ or are given by

$$a'_{ij} = \begin{cases} a_{ij} - v_n & \text{for } j = \ell, \\ a_{ij} + v_n & \text{for } j = k, \\ a_{ij} & \text{otherwise;} \end{cases}$$

hence $\sum_j a'_{ij} = \sum_j a_{ij}$.

Lemma 2. X is a maximal flow.

Proof. At the point where termination occurs, we have defined a set S of nodes, consisting of those nodes $P_i$ for which $v_i, \mu_i$ have been defined; and further, $P_1 \in S$, $P_n \notin S$. Consider the set $\Gamma$ of directed arcs $P_iP_j$ such that $P_i \in S$, $P_j \notin S$.* Clearly $a_{ij} = 0$ for such pairs i,j, as otherwise we would have defined $v_j, \mu_j$.

We will show that $\Gamma$ is a cut whose value is equal to $\sum_j x_{1j}$, thus proving that X is a maximal flow and $\Gamma$ a minimal cut.

That $\Gamma$ is a cut is clear. For if there were a chain $P_1P_{i_1}\ldots P_{i_k}P_n$ with the arcs $P_1P_{i_1} \notin \Gamma, \ldots, P_{i_k}P_n \notin \Gamma$, then we could deduce successively (since $P_1 \in S$) that $P_{i_1} \in S, \ldots, P_n \in S$, a contradiction. To see that $\Gamma$ has value equal to the flow value, notice that

$$\sum_j (c_{ij} - a_{ij}) = \begin{cases} 0 & (1 < i < n), \\ \sum_j x_{1j} & (i = 1) \end{cases} .$$

Now sum these equations over those i for which $P_i \in S$. On the left side, if $P_i$ and $P_j$ are both in S, then $c_{ij} - a_{ij}$ and $c_{ji} - a_{ji}$ are both in the summation and are negatives of each other. All that remains are terms of the form $c_{ij} - a_{ij}$, where $P_i \in S$, $P_j \notin S$. For these, as we pointed out above, $a_{ij} = 0$.

* The set $\Gamma$ is actually the set of "left arcs" defined in [5].

Thus the sum on the left reduces precisely to the sum of capacities of members of $\Gamma$, as was to be shown.

We append a simple example, using the matrix format instead of a picture of the network, to illustrate the computation in this form.

__Example__.

$$A_1 = C = \begin{array}{c} \\ \left[\begin{array}{ccccc} & 4 & 2 & 2 & 0 \\ 0 & & 4 & 2 & 2 \\ 0 & 4 & & 3 & 1 \\ 0 & 2 & 2 & & 4 \\ 0 & 0 & 0 & 0 & \end{array}\right] \end{array} \quad \begin{array}{cc} v & \mu \\ \infty & 0 \\ 4 & 1 \\ 2 & 1 \\ 2 & 1 \\ 2 & 2 \end{array}$$

$$A_2 = \left[\begin{array}{ccccc} & 2 & 2 & 2 & 0 \\ 2 & & 4 & 2 & 0 \\ 0 & 4 & & 3 & 1 \\ 0 & 2 & 2 & & 4 \\ 0 & 2 & 0 & 0 & \end{array}\right] \quad \begin{array}{cc} \infty & 0 \\ 2 & 1 \\ 2 & 1 \\ 2 & 1 \\ 1 & 3 \end{array}$$

$$A_3 = \left[\begin{array}{ccccc} & 2 & 1 & 2 & 0 \\ 2 & & 4 & 2 & 0 \\ 1 & 4 & & 3 & 0 \\ 0 & 2 & 2 & & 4 \\ 0 & 2 & 1 & 0 & \end{array}\right] \quad \begin{array}{cc} \infty & 0 \\ 2 & 1 \\ 1 & 1 \\ 2 & 1 \\ 2 & 4 \end{array}$$

$$A_4 = \left[\begin{array}{ccccc} & 2 & 1 & 0 & 0 \\ 2 & & 4 & 2 & 0 \\ 1 & 4 & & 3 & 0 \\ 2 & 2 & 2 & & 2 \\ 0 & 2 & 1 & 2 & \end{array}\right] \quad \begin{array}{cc} \infty & 0 \\ 2 & 1 \\ 1 & 1 \\ 2 & 2 \\ 2 & 4 \end{array}$$

$$A_5 \quad \left[\begin{array}{ccccc} & 0 & 1 & 0 & 0 \\ 4 & & 4 & 0 & 0 \\ 1 & 4 & & 3 & 0 \\ 2 & 4 & 2 & & 0 \\ 0 & 2 & 1 & 4 & \end{array}\right] \quad \begin{array}{cc} \infty & 0 \\ 1 & 3 \\ 1 & 1 \\ 1 & 3 \\ & \end{array}$$

Thus the computation terminates with $A_8$, giving a total flow of 7. The minimal cut comprises the arcs 15, 25, 35, 45, with value 7.

## 4. THE HITCHCOCK PROBLEM

The Hitchcock transportation problem is perhaps one of the "most solved" linear programming problems in existence. We shall propose yet another computation for the problem which will amount to solving a sequence of flow problems of a particularly simple kind. The basic idea, which stems from a proof given by Egervary [4] for a theorem of Konig [6, p. 232] on linear graphs, has been used by Kuhn [7] to develop a very efficient combinatorial algorithm for the optimal assignment problem, a special case of the Hitchcock problem. Our method differs only in details from the Kuhn algorithm in this case.

We take the Hitchcock problem in the following form: Given a matrix $D = (d_{ij})$ of nonnegative integers, and two sets of nonnegative integers $(a_1,\ldots,a_m)$, $(b_1,\ldots,b_n)$, with $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$, it is desired to find a matrix $X = (x_{ij})$ satisfying the constraints

$$(8)\qquad \begin{aligned} &x_{ij} \geq 0, \\ &\sum_j x_{ij} = a_i \ , \\ &\sum_i x_{ij} = b_j \end{aligned}$$

which minimizes the linear form

$$(9)\qquad \sum_{i,j} d_{ij}\, x_{ij} \ .$$

A physical interpretation of the problem is that there are m originating points for a commodity, the i th point having $a_i$ units, and n destinations, the j th one requiring $b_j$ units. If $d_{ij}$ is the cost, per unit of commodity, of shipping from origin i to destination j, find a shipping program of minimal cost.

The dual of the Hitchcock problem is: Find $\alpha_i$, $\beta_j$ satisfying the constraints

(10) $$\alpha_i + \beta_j \leq d_{ij} \qquad (i = 1,\ldots,m;\ j = 1,\ldots,n)$$

which maximize

(11) $$\sum_i a_i \alpha_i + \sum_j b_j \beta_j .$$

The proof that the algorithm to be described yields a solution to the Hitchcock problem will be based on the fact that the dual form (11) increases by at least one unit with the solution of each successive flow problem.

Each of the flow problems will be of the following form. Find $X = (x_{ij})$ satisfying

(12) $$\begin{cases} x_{ij} \geq 0 , \\ \sum_j x_{ij} \leq a_i , \\ \sum_i x_{ij} \leq b_j , \\ \quad x_{ij} = 0 \text{ for a given set } \Omega \text{ of pairs } i,j, \end{cases}$$

which maximizes

(13) $$\sum_{i,j} x_{ij} .$$

This is actually a special Hitchcock problem. To see that it is also a flow problem, set up the directed network of m + n + 2 nodes shown in Fig. 6,

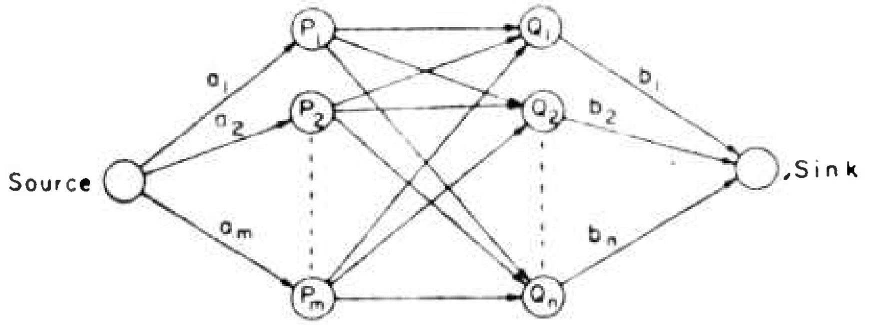

Fig. 6

where the capacity on the directed arc $P_iQ_j$ is zero if $i,j \epsilon \Omega$ , large otherwise, the capacities of source and sink arcs are the $a_i$ and $b_j$, as shown, and interpret $x_{ij}$ as the flow from $P_i$ to $Q_j$.

To solve such a problem we may of course use the computation of Section 3 involving, in this case, an m + n + 2 by m + n + 2 matrix. It is possible (and computationally convenient) to describe the process in terms of an m by n array. The verification that the two descriptions are the same for this particular class of problems will be left to the reader.

Let X be a solution of the constraints (12). Corresponding to certain of the rows i = 1,...,m of X we will define integers $v_i, \mu_i$; similarly, for certain of the columns j = 1,...,n we will define integers $w_j$, $\lambda_j$. These definitions will be made recursively; first a set of $v_i$, $\mu_i$ will be defined, from these a set of $w_j$, $\lambda_j$ will be defined, and so forth, alternating between the rows and columns.

For those i such that $\sum_j x_{ij} < a_i$, define

(14) $$v_i = a_i - \sum_j x_{ij}, \mu_i = 0 .$$

Next select an i which has been labeled and scan for all (unlabeled) j such that $i,j \notin \Omega$; for these j, define

(15) $$w_j = v_i, \lambda_j = i.$$

Repeat until the previously labeled i's are exhausted. We then select a labeled j and scan for unlabeled i such that $x_{ij} > 0$; for these i, define

(16) $$v_i = \min (x_{ij}, w_j), \mu_i = j.$$

Repeat until the previously labeled j's are exhausted. Again select one of the i's just labeled, look for unlabeled j such that $i,j \notin \Omega$, and define $w_j$, $\lambda_j$ by (15). Continue in this fashion, using (15) and (16) alternately until either $w_j$, $\lambda_j$ have been defined for some j with $\sum_j x_{ij} < b_j$, or until no further definitions may be made. In the latter case X is maximal; in the former we can get an improvement as follows. Let

(17) $$v = \min (w_j, b_j - \sum_i x_{ij}) .$$

Alternately add and subtract v from the sequence

(18) $$x_{\lambda_j j},\ x_{\lambda_j j_1},\ x_{i_1 j_1},\ x_{i_1 j_2}, \ldots, x_{i_{k-1} j_k}, x_{i_k j_k}\ ,$$

where $j_1 = \mu_{\lambda_j}$, $i_1 = \lambda_{j_1}$, $j_2 = \mu_{i_1}$, $i_2 = \lambda_{j_2}, \ldots, j_k = \mu_{i_{k-1}}$,

$i_k = \lambda_{j_k}$, and $\mu_k = 0$.

<u>Example.</u>

$j$

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | $a_i$ | $v_i$ | $\mu_i$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | ① | ① | | | | | | | 2 | 1 | 3 |
| 2 | | | | | | | ② | | ① | | 3 | | |
| 3 | | ① | | | | | | | | ○ | 1 | 1 | 2 |
| 4 | ① | | | | ③ | | | | | | 4 | 1 | 1 |
| 5 | | ○ | | | ③ | | ○ | ○ | | ○ | 3 | | |
| 6 | | ○ | ○ | | | | | ② | | | 3 | 1 | 0 |
| 7 | ○ | ○ | | | | | | | ① | | 1 | 1 | 9 |
| 8 | | ○ | | ① | | ○ | | | | | 1 | 1 | 4 |
| 9 | | | | | | | | ○ | ○ | ③ | 5 | 2 | 0 |
| 10 | | | | ⑥ | | ② | | | | | 8 | 1 | 4 |
| $b_j$ | 1 | 1 | 1 | 8 | 9 | 2 | 2 | 2 | 2 | 3 | | | |
| $w_j$ | 1 | 1 | 1 | 1 | 1 | 1 | | 2 | 2 | 2 | | | |
| $\lambda_j$ | 7 | 6 | 6 | 1 | 4 | 10 | | 9 | 9 | 9 | | | |

Cells containing a circle comprise $\bar{\Omega}$ (the complement of $\Omega$). The entries within the circles (zeros elsewhere) constitute an X satisfying the constraints (12). The defining process terminates with $w_5 = 1$, $\lambda_5 = 4$, since column 5 is a column in which the sum of the entries $x_{i5}$ is less than $b_5$; i.e., $3 + 3 < 9$.

The sequence along which an improvement of min $(w_s, b_s - \sum_i x_{is}) = 1$ can be made is $x_{4s} = 3$, $x_{41} = 1$, $x_{71} = 0$, $x_{7s} = 1$, $x_{ss} = 0$, as is easily read off using the $\lambda$'s and $\mu$'s alternately.

We are now in a position to describe a general routine for the Hitchcock problem. As a starting point, form the difference matrix

(19) $$d_{ij} - \alpha_i - \beta_j ,$$

where $\alpha_i = \min_j d_{ij}$, $\beta_j = \min_i (d_{ij} - \alpha_i)$. Thus $d_{ij} - \alpha_i - \beta_j \geq 0$; i.e., $\alpha_i$, $\beta_j$ satisfy the dual constraints (10). The next step is to solve the flow problem with

(20) $$\Omega = \{ij \mid d_{ij} - \alpha_i - \beta_j > 0\} .$$

If the maximizing flow X satisfies $\sum_{i,j} x_{ij} = \sum_i a_i$, then X is a minimizing solution to the original Hitchcock problem.[s] If, on the

[s] This is easily deduced as follows. Observe first of all that for any $\alpha_i$, $\beta_j$, the two Hitchcock problems with cost matrices $d_{ij}$ and $d_{ij} - \alpha_i - \beta_j$ are equivalent; for $\sum_j x_{ij} = a_i$, $\sum_i x_{ij} = b_j$ imply that $\sum_{i,j} (d_{ij} - \alpha_i - \beta_j) x_{ij} = \sum_{i,j} d_{ij} x_{ij}$ $- \sum_i a_i \alpha_i - \sum_j b_j \beta_j$, and the last two sums on the right are independent of $x_{ij}$. Thus if we have $\alpha_i$, $\beta_j$ with $d_{ij} - \alpha_i - \beta_j \geq 0$, and are able to find an X satisfying (8) and $x_{ij} = 0$ for $ij \in \Omega$, then clearly X minimizes $\sum_{i,j} (d_{ij} - \alpha_i - \beta_j) x_{ij}$, hence minimizes $\sum_{i,j} d_{ij} x_{ij}$.

other hand, $\sum_{i,j} x_{ij} < \sum_I a_i$, let I be the index set of the labeled rows of X, J the index set of the labeled columns, and define new dual variables by

$$(21) \quad \alpha_i' = \begin{cases} \alpha_i + k & (i \in I), \\ \alpha_i & (i \notin I), \end{cases} \qquad \beta_j' = \begin{cases} \beta_j - k & (j \in J), \\ \beta_j & (j \notin J), \end{cases}$$

where $k = \min_{\substack{i \in I \\ j \notin J}} (d_{ij} - \alpha_i - \beta_j)$. Notice that $k > 0$, since pairs $i,j$ with $i \in I$, $j \notin J$ are contained in $\Omega$.

<u>Lemma</u> 3. $\sum_I a_i \alpha_i' + \sum_J b_j \beta_j' > \sum_I a_i \alpha_i + \sum_J b_j \beta_j$.

<u>Proof</u>. The fact that X is a maximal flow implies that $\sum_I x_{ij} = b_j$ for $j \in J$. Also, the labeling process ensures that $x_{ij} = 0$ for $i \notin I$, $j \in J$, and, as we have mentioned earlier, $x_{ij} = 0$ for $i \in I$, $j \notin J$. Since all i with $\sum_J x_{ij} < a_i$ are in I, it follows that

$$\sum_{i \in I} a_i > \sum_{i \in I} \sum_{j \in J} x_{ij} = \sum_{j \in J} b_j ,$$

and hence the new dual form has been increased by the amount

$$k\left( \sum_{i \in I} a_i - \sum_{j \in J} b_j \right) > 0 .$$

Another way to see this is to note that the minimal cut in the associated network (Fig. 6) has value $\sum_{i \notin I} a_i + \sum_{j \in J} b_j < \sum_{I} a_i$, and hence $\sum_{i \in I} a_i > \sum_{j \in J} b_j$ .

Now form the new difference matrix $d_{ij} - \alpha_i - \beta_j \geq 0$ by subtracting k from the I-rows of the previous difference matrix, and adding k to the J-columns. We may now take the maximal flow X of the previous flow problem as a starting point in the new flow problem and proceed as before.

## REFERENCES

1. Boldyreff, Alexander W., "Determination of the Maximal Steady State Flow of Traffic Through a Railroad Network," The RAND Corporation, Paper P-687, August 5, 1955.

2. Dantzig, G. B., and D. R. Fulkerson, On the Max Flow Min Cut Theorem of Networks, The RAND Corporation, Research Memorandum RM-1418-1, January 1, 1955.

3. Dantzig, G. B., and D. R. Fulkerson, "Computation of Maximal Flows in Networks," The RAND Corporation, Paper P-677, April 1, 1955.

4. Egerváry, E., "Matrixok Kombinatorius Tulajdonsagairól," Matematikai es Fizikai Lapok, Vol. 38, 1931, pp. 16-28; translated by H. W. Kuhn as Paper 4, Issue 11 of Logistics Papers, George Washington University Logistics Research Project, 1955.

5. Ford, L. R., Jr., and D. R. Fulkerson, "Maximal Flow Through a Network," The RAND Corporation, Paper P-605, November 19, 1954.

6. König, Dénes, Theorie der endlichen und unendlichen Graphen, Chelsea Publishing Company, New York, 1950.

7. Kuhn, H. W., "A Combinatorial Algorithm for the Assignment Problem," Issue 11 of Logistics Papers, George Washington University Logistics Research Project, 1955.

8. Robacker, J. T., On Network Theory, The RAND Corporation, Research Memorandum RM-1498, May 26, 1955.